CS 3710 Introduction to Cybersecurity
Term: Fall 2025

Lab Exercise 1 – Introduction to Password Cracking

Due Date: September 5, 2025 11:59pm
Points Possible: 7 points

Name: Laxmi Ghanate

By submitting this assignment you are digitally signing the honor code, "On my honor, I pledge that I have neither given nor received help on this assignment."

AI assistance is permitted on this assignment. Please be aware that AI answers are not always correct, so validate the answer. If you are opposed to using AI, you do not need to do so on the questions specifically requesting AI. Please cite all sources including AI.

1. Overview

This lab exercise will provide some hands-on experience with password strength analysis using command-line tools in Linux.

2. Resources required

This exercise requires a Kali Linux VM running in the Virginia Cyber Range.

3. Initial Setup

From your Virginia Cyber Range course, select the **Cyber Basics** environment. Click "Start My Environment" to start your environment and once it is ready click "Join My Environment" to open your Linux desktop.

4. Tasks

Task 1: Introduction to password auditing.

On Linux systems, user accounts are stored in the **`/etc/passwd`** file (world-readable text file) and passwords are hashed and stored in **`/etc/shadow`** (a text file only readable by root). Click on the Terminal Emulator to open a command prompt. You will need to become an administrator on the system to see the shadow file. Type "**`sudo su -`**" and hit enter. You will notice your command prompt changed from a **$** to a **#** and your user changed from student to root. Go ahead and "cat" those two password files to see what they look like.

Question #1: What hash type is used by your Cyber Range version of Linux? How can you determine that by looking at the hashed passwords in /etc/shadow? *(.5 point)*

By looking at the password entry for student user in /etc/shadow: "student:yj9T$ZMaX6IBoXlrSXMssii.Sl0$V1Upn0z0INYW4o0/z/jzoo6fDJdWAxw3Wf5Pt/N6Pn9:20329:0:99999:7:::" we can see that the prefix identified "y" that tells us the hashing algorithm. This would mean that it's yescrypt.

Question #2: What are two other hash IDs and their types that you may see in an /etc/shadow file? (The ID is the numbers/letters that identify the hash and the type is the name of the hash) Give some details on each algorithm. (.5 point)

Two other hash IDs we may see are 5, SHA-256, and 6, SHA-512. SHA-256 generates a 256 bit hash while SHA-512 generates a 512-bit hash which makes it stronger against brute force attacks.

*Question #3: Use an AI tool to explain password salting and why it is important. Paste the answer here. Then, use AI to find a **real-world breach example** where password hashing was involved. Summarize the case and explain how salting would have changed the outcome. Cite your sources. (.5 point)*

According to ChatGPT version 5, password salting is the process of appending unique, random data to passwords prior to hashing, so that even identical passwords yield different hash outputs—this prevents attackers from leveraging precomputed lists like rainbow tables. A real-world example is the 2012 LinkedIn breach, where weak, unsalted SHA-1 hashes were cracked en masse; had salting been implemented, the breach would have been far harder to exploit because each hash would require individual, much more time-consuming cracking efforts.

https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/?utm

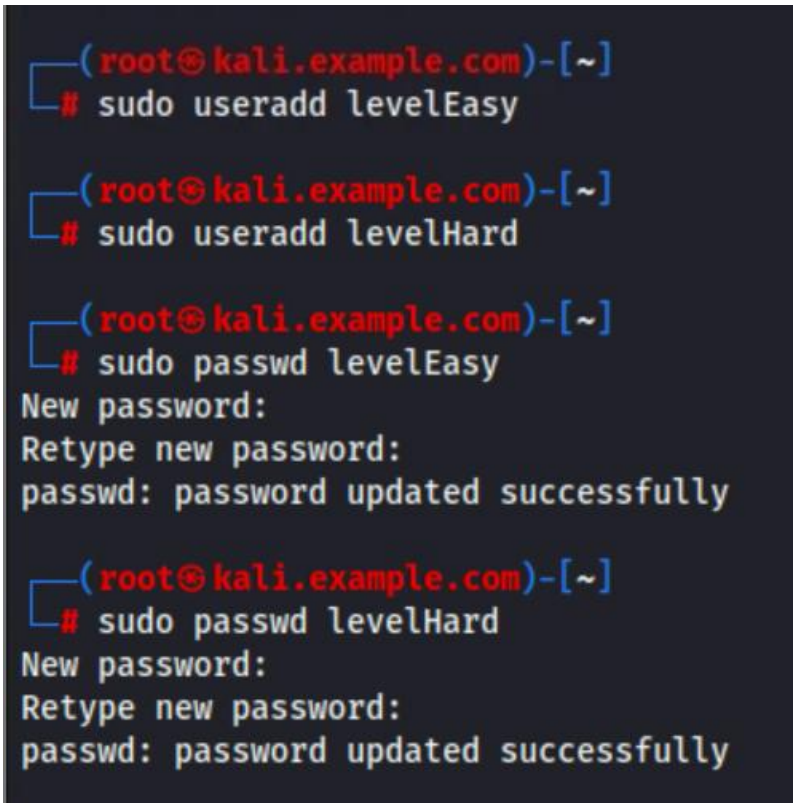
https://arxiv.org/abs/1703.06586?utm

We'll use a password auditing tool called John the Ripper (JTR), a very effective and widely known password cracker. JTR is available from www.openwall.com/john. JTR is already installed in the virtual environment so you won't need to download it.

Task 2: **Crack Linux passwords.**

1. Create 2 new accounts, one with an easy to guess password (such as 1234) and one with a difficult to guess password.

Question #4: Cut and paste or screen capture the commands you used to create the accounts and set the passwords. *(.5 point)*



2. Now let's see which ones we can crack. Run john against the /etc/shadow file. You will need to use the **`-format:crypt`** command line option to crack this particular hash method.

JTR will attempt to crack the passwords and display any that it 'cracks' as it goes along. It starts in "single crack" mode, mangling username and other account information. It then moves on to a dictionary attack using a default dictionary, then with a hybrid attack, then brute force where it will try every possibly combination of characters (letters, numbers, and special characters) until it cracks them all. You may see several warnings about candidates buffered for the current salt and that is ok. You can ignore those warnings.

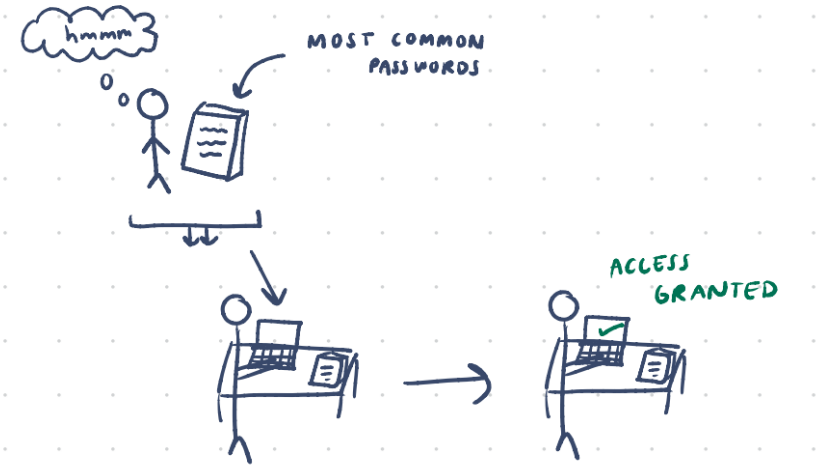
The account with the easy to guess password should be cracked rather quickly. Wait for a little bit for it to crack the difficult password, but don't wait too long as it could take months or years to complete if your password is really strong! Press [CTRL]-[C] to stop execution if it doesn't automatically complete and return to the command prompt.

Question #5: Provide a screenshot of your JTR cracked passwords (.5 point)

```
┌──(root㉿kali.example.com)-[~]
└─# sudo cp /etc/shadow /home/student/shadow-copy
sudo cp /etc/passwd /home/student/passwd-copy

┌──(root㉿kali.example.com)-[~]
└─# sudo unshadow /home/student/passwd-copy /home/student/shadow-copy > /home/student/unshadow
ed.txt
Created directory: /root/.john

┌──(root㉿kali.example.com)-[~]
└─# john --format=crypt /home/student/unshadowed.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0
for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
student          (student)
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 54 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst
1g 0:00:03:19 4.88% 2/3 (ETA: 22:20:25) 0.005004g/s 40.83p/s 72.42c/s 72.42C/s Pete..Overkill
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

┌──(root㉿kali.example.com)-[~]
└─# john --show /home/student/unshadowed.txt
student:student:1000:1001:Cyber Range Student:/home/student:/bin/bash

1 password hash cracked, 1 left
```

Question #6: Briefly describe how a dictionary-based password attack works. Then, turn that explanation into an illustrated and labeled visual representation. (.75 point)

A dictionary-based password attack is when an attacker uses a precompiled list of common passwords and tests them against the login system. If the password matches an entry, access is gained without testing every possible combination.



Apologies for the awful drawing but it shows a person, named Joe, who gains access to a book with the Most Common Passwords and then uses it to gain access to a system. He goes through many accounts with the same passwords to see if it is able to unlock any of them.

Question #7: Briefly describe how a brute force password attack works. Then, write your own ***analogy*** that explains the concept to a non-technical friend.. *(.75 point)*

A brute force password attack works when every possible character combo is used until the correct password is found. This is similar to guessing a lock code by starting at 0000 and working your way up to 9999 and trying every number until it opens.

John uses the following files to manage execution. Most are all stored in the **`/usr/share/john`** folder on your Kali virtual machine (john.pot is stored elsewhere as indicated):

- **`password.lst`** is john's default dictionary. You can **cat** this file to look at it. You can specify another wordlist on the command line using the **`--wordlist=`** directive (for example **`# john --wordlist=/usr/share/dict/american-english /etc/shadow`**
- **`john.conf`** is read when JTR starts up and has rules for dictionary mangling for the hybrid crack attempt
- **`john.rec`** is used to record the status of the current password cracking attempt. If john crashes, it will start where it left off instead of starting again from the beginning of the dictionary.
- **/root/.john/`john.pot`** lists passwords that have already been cracked. If you run john again on the same shadow file, it won't show these cracked passwords unless you delete this file first using **rm /root/.john/john.pot.**

Task 3. More password cracking.

John the Ripper's default dictionary is a short list of common passwords. Sometimes a standard English dictionary is a better option. In this exercise we will 1) download a new Linux shadow file that contains a set of user accounts and hashed passwords, 2) download a different dictionary, and then 3) attempt to determine the passwords using the default dictionary and the new dictionary.

1. Download the following new shadow file using the wget command:

   `artifacts.virginiacyberrange.net/gencyber/shadow`

2. Take a look at the newly downloaded shadow file. It should be in your current working directory, it is not /etc/shadow. Notice that it uses a different hash ID. You won't need to use the -format:crypt command line option, let's see if John can automatically figure out the hash. Run John against the newly downloaded shadow file. Let John run for a few minutes, then stop with [CTRL]-[C].

*Question #8: Which passwords are revealed**?*** *(cut and paste or screen capture) (.5 point)*

```
┌──(root㉿kali.example.com)-[~]
└─# cat shadow
user1:$6$Pj5MJMN9$9It7gUS1qhRImPehIXxCNewJRujKmU1hPeLSwuuwHSyWLfGVf17I5aVyK0x0O0r//trrfWs
:
user2:$6$y1cRLF4J$oEAzmCxjgF4rUO8NpjH.WNuqrGrR7YToHvwa9NB9j0zcaz/CaZK/0oOn/8SQk0b5W.EpiFo
:
user3:$6$4VBMoPws$VAbtOxCdcWl6XBOpQTyiRuZqeMNNYL0ih8fGNEkWhTfI/6YIhrcEY6uiQ8SOKllSzX0vlsr
:
user4:$6$YI0/GDkI$TdAbv3B7XuzA1k4LNnt/X.LXZzD8AEzR1QG59ge5asQ1ELG8oJuWLPPF/PADLDxxUClRfEZ
:
┌──(root㉿kali.example.com)-[~]
└─# john shadow
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (sha512crypt, crypt(3) $6$ [SHA512 512/51
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Africa           (user3)
Adams            (user2)
Proceeding with incremental:ASCII
2g 0:00:05:51  3/3 0.005694g/s 1287p/s 2959c/s 2959C/s clev05..cliaha
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

┌──(root㉿kali.example.com)-[~]
└─# john --show shadow
user2:Adams:17319:0:99999:7:::
user3:Africa:17319:0:99999:7:::

2 password hashes cracked, 2 left

┌──(root㉿kali.example.com)-[~]
└─#
```

3. Next we will run John again with a different dictionary. First, we will download an American English dictionary on to our Kali Linux system. (If the dictionary is already there, it will be updated.)

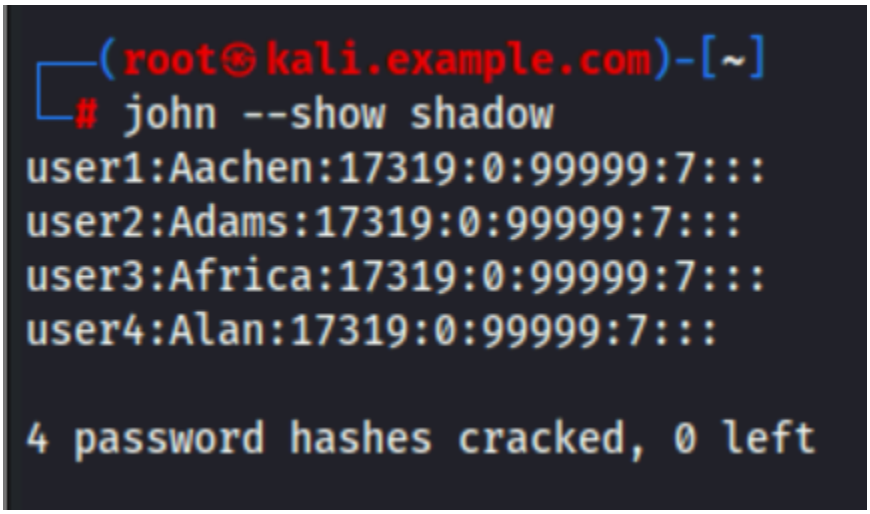
```
# sudo apt update; sudo apt install -y wamerican
```

4. Clear the John cache from the previous run by deleting the **/root/.john/john.pot** file.

5. Next run John against the downloaded shadow file again but this time using the newly downloaded dictionary by invoking the --wordlist option at the command line with the location of the new dictionary (**`--wordlist=/usr/share/dict/american-english`**)

   Note: If you get an error about a locked /root/.john/john.rec file, you can delete that file.

Question #9: Which passwords were revealed this time? *(cut and paste or screen capture) (.5 point)*



Question #10: What is the difference between the two dictionaries that made one attempt more effective than the other? (Be specific. You may want to take a look at each of the dictionaries or metadata about the dictionaries to compare them.) (1 point)

The default password.lst dictionary is small and contains only the most common passwords, so it only cracked a few accounts. The American English dictionary is much larger and includes thousands of real words and names, which allowed it to successfully crack all the password hashes since they were dictionary words.

*Question #11: Use an AI tool to explain additional methods that will help provide more **secure authentication** and protect against password cracking. Paste the response for two methods here. Then, consider these methods for a **specific system you use daily or weekly**. Describe how these protections work in that system and why they are effective. (Note: Please do not include the method of longer and/or more complex passwords – let's go beyond that.) (1 point)*

Method 1: Multi-Factor Authentication (MFA)

ChatGPT (version 5) explains that MFA requires users to provide two or more forms of identification, such as a password plus a one-time code sent to their phone, or a biometric factor like a fingerprint. Even if a password is stolen or cracked, the attacker cannot log in without the second factor.
Example: I use MFA on my bank account and university account occasionally. After entering my password, I must confirm a code from my phone, often from Duo Push. This ensures that even if someone guesses or cracks my password, they still cannot access my account without my device.

Method 2: Account Lockout and Rate Limiting
ChatGPT (version 5) explains that these mechanisms limit repeated failed login attempts by temporarily locking accounts cannot or slowing responses. This makes brute force and dictionary attacks impractical, as attackers test large numbers of passwords quickly.
Example: My Instagram account locks itself if there's more than 5 failed repeated log in attempts. Unfortunately, this has locked me out more than once.

To close the exercise, just click the X on the terminal window to close it and click on the Log Out icon in the upper right hand corner of the screen to log out.

By submitting this assignment you are digitally signing the honor code, "I pledge that I have neither given nor received help on this assignment".

END OF EXERCISE

References

- John the Ripper (JTR): www.openwall.com/john